On amending the Maskin's sufficiency theorem by using complex numbers

Haoyang Wu *

Abstract

The Maskin's theorem is a fundamental work in the theory of mechanism design. In this paper, we will propose a self-enforcing agreement by which agents can fight a bad social choice rule which satisfies monotonicity and no-veto if an additional condition is satisfied. The agreement is based on complex numbers and is justified if the designer receives messages from the agents through some communication channels (e.g., Internet). Under the assumption of complete information among agents, the designer cannot prevent the agents from signing such agreement. Thereby, the Maskin's sufficiency theorem is amended.

Key words: Mechanism design; Nash implementation.

1 Introduction

Nash implementation is the cornerstone of the mechanism design theory. The Maskin's theorem provides an almost complete characterization of social choice rules (SCRs) that are Nash implementable [1]. Because an SCR is specified by a designer, a desired outcome from the designer's perspective may not be desirable for the agents. However, when the number of agents are at least three, the designer can always implement an SCR which satisfies monotonicity and no-veto in Nash equilibrium even if all agents dislike it (See Table 1 in Section 3.1).

With the development of network economics, it is not unusual that the designer receives messages from agents through some communication channel (e.g., Internet). For these cases, we will propose a self-enforcing agreement by which

* Wan-Dou-Miao Research Lab, Suite 1002, 790 WuYi Road, Shanghai, 200051, China.

Email address: `hywch@mail.xjtu.edu.cn`, `Tel: 86-18621753457` (Haoyang Wu).

agents can fight a bad SCR when they face the Maskin's mechanism, even if the SCR satisfies monotonicity and no-veto. Put differently, the Maskin's sufficiency theorem will be amended for such cases.

The rest of the paper is organized as follows: Section 2 recalls preliminaries of the mechanism design theory [2]; Section 3 is the main part of this paper, where we will propose an agreement using complex numbers to amend the sufficiency part of the Maskin's theorem. Section 4 draws conclusions.

2 Preliminaries

Let $N = \{1, \cdots, n\}$ be a finite set of *agents* with $n \geq 2$, $A = \{a_1, \cdots, a_k\}$ be a finite set of social *outcomes*. Let T_i be the finite set of agent i's types, and the *private information* possessed by agent i is denoted as $t_i \in T_i$. We refer to a profile of types $t = (t_1, \cdots, t_n)$ as a *state*. Let $\mathcal{T} = \prod_{i \in N} T_i$ be the set of states. At state $t \in \mathcal{T}$, each agent $i \in N$ is assumed to have a complete and transitive *preference relation* $\succeq_i^t$ over the set A. We denote by $\succeq^t = (\succeq_1^t, \cdots, \succeq_n^t)$ the profile of preferences in state t, and denote by $\succ_i^t$ the strict preference part of $\succeq_i^t$. Fix a state t, we refer to the collection $E =< N, A, (\succeq_i^t)_{i \in N} >$ as an *environment*. Let ε be the class of possible environments. A *social choice rule* (SCR) F is a mapping $F : \varepsilon \rightarrow 2^A \backslash \{\emptyset\}$. A *mechanism* $\Gamma = ((M_i)_{i \in N}, g)$ describes a message or strategy set M_i for agent i, and an outcome function $g : \prod_{i \in N} M_i \rightarrow A$. M_i is unlimited except that if a mechanism is direct, $M_i = T_i$.

An SCR F satisfies *no-veto* if, whenever $a \succeq_i^t b$ for all $b \in A$ and for all agents i but perhaps one j, then $a \in F(E)$. An SCR F is *monotonic* if for every pair of environments E and E', and for every $a \in F(E)$, whenever $a \succeq_i^t b$ implies that $a \succeq_i^{t'} b$, there holds $a \in F(E')$. We assume that there is *complete information* among the agents, i.e., the true state t is common knowledge among them. Given a mechanism $\Gamma = ((M_i)_{i \in N}, g)$ played in state t, a *Nash equilibrium* of Γ in state t is a strategy profile m^* such that: $\forall i \in N, g(m^*(t)) \succeq_i^t g(m_i, m_{-i}^*(t)), \forall m_i \in M_i$. Let $\mathcal{N}(\Gamma, t)$ denote the set of Nash equilibria of the game induced by Γ in state t, and $g(\mathcal{N}(\Gamma, t))$ denote the corresponding set of Nash equilibrium outcomes. An SCR F is *Nash implementable* if there exists a mechanism $\Gamma = ((M_i)_{i \in N}, g)$ such that for every $t \in \mathcal{T}$, $g(\mathcal{N}(\Gamma, t)) = F(t)$.

Maskin [1] provided an almost complete characterization of SCRs that were Nash implementable. The main results of Ref. [1] are two theorems: 1) (*Necessity*) If an SCR is Nash implementable, then it is monotonic. 2) (*Sufficiency*) Let $n \geq 3$, if an SCR is monotonic and satisfies no-veto, then it is Nash implementable. In order to facilitate the following investigation, we briefly recall

the Maskin's mechanism published in Ref. [2] as follows:

Consider the following mechanism $\Gamma = ((M_i)_{i\in N}, g)$, where agent i's message set is $M_i = A \times \mathcal{T} \times \mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of non-negative integers. A typical message sent by agent i is described as $m_i = (a_i, t_i, z_i)$. The outcome function g is defined in the following three rules: (1) If for every agent $i \in N$, $m_i = (a, t, 0)$ and $a \in F(t)$, then $g(m) = a$. (2) If $(n-1)$ agents $i \neq j$ send $m_i = (a, t, 0)$ and $a \in F(t)$, but agent j sends $m_j = (a_j, t_j, z_j) \neq (a, t, 0)$, then $g(m) = a$ if $a_j \succ_j^t a$, and $g(m) = a_j$ otherwise. (3) In all other cases, $g(m) = a'$, where a' is the outcome chosen by the agent with the lowest index among those who announce the highest integer.

3 An agreement to amend the Maskin's sufficiency theorem

This section is the main part of this paper. In the beginning, we will show a bad SCR which satisfies monotonicity and no-veto. It is Nash implementable although all agents dislike it. Then, we will define some matrices and propose an agreement with complex numbers, by which the agents can amend the Maskin's sufficiency theorem.

3.1 An example

Table 1: A bad SCR that is monotonic and satisfies no-veto.

State t^1			State t^2		
Apple	*Lily*	*Cindy*	*Apple*	*Lily*	*Cindy*
a^3	a^2	a^1	a^4	a^3	a^1
a^1	a^1	a^3	a^1	a^1	a^2
a^2	a^4	a^2	a^2	a^2	a^3
a^4	a^3	a^4	a^3	a^4	a^4
$F(t^1) = \{a^1\}$			$F(t^2) = \{a^2\}$		

Let $N = \{Apple, Lily, Cindy\}$, $\mathcal{T} = \{t^1, t^2\}$, $A = \{a^1, a^2, a^3, a^4\}$. In each state $t \in \mathcal{T}$, the preference relations $(\succeq_i^t)_{i\in N}$ over the outcome set A and the corresponding SCR F are given in Table 1. The SCR F is bad from the viewpoint of the agents because in state $t = t^2$, all agents unanimously prefer a Pareto-efficient outcome $a^1 \in F(t^1)$: for each agent i, $a^1 \succ_i^{t^2} a^2 \in F(t^2)$.

It seems that in state $t = t^2$, $(a^1, t^1, 0)$ should be a unanimous m_i for each

agent i. As a result, a^1 may be generated by rule 1. However, *Apple* has an incentive to unilaterally deviate from $(a^1, t^1, 0)$ to $(a^4, *, *)$, since $a^1 \succ_{Apple}^{t^1} a^4$, $a^4 \succ_{Apple}^{t^2} a^1$; *Lily* also has an incentive to unilaterally deviate from $(a^1, t^1, 0)$ to $(a^3, *, *)$, since $a^1 \succ_{Lily}^{t^1} a^3$, $a^3 \succ_{Lily}^{t^2} a^1$.

Note that either *Apple* or *Lily* can certainly obtain her expected outcome only if just one of them deviates from $(a^1, t^1, 0)$ (If this case happened, rule 2 would be triggered). But this condition is unreasonable, because all agents are rational, nobody is willing to give up and let the others benefit. Therefore, both *Apple* and *Lily* will deviate from $(a^1, t^1, 0)$. As a result, rule 3 will be triggered. Since *Apple* and *Lily* both have a chance to win the integer game, the winner is uncertain. Consequently, the final outcome is uncertain between a^3 and a^4.

To sum up, although every agent prefers a^1 to a^2 in state $t = t^2$, a^1 cannot be yielded in Nash equilibrium. Indeed, the Maskin's mechanism makes the Pareto-inefficient outcome a^2 be implemented in Nash equilibrium in state $t = t_2$.

Can the agents find a way to let the Pareto-inefficient outcome a^2 not be Nash implemented in state $t = t^2$ when they face the Maskin's mechanism? Interestingly, we will show that the answer may be "yes". To do so, a new weapon - the complex number - will be used. Although it has been well-known for hundreds of years, it has never been used in the theory of mechanism design. In what follows, first we will define some matrices, then we will propose an agreement to break through the Maskin's sufficiency theorem.

3.2 Definitions

Definition 1: Let $\hat{I}, \hat{\sigma}$ be two 2×2 matrices. $\overrightarrow{C}, \overrightarrow{D}$ are defined as two basic vectors:

$$\hat{I} \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \hat{\sigma} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \overrightarrow{C} \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \overrightarrow{D} \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \qquad (1)$$

Hence, $\hat{I}\overrightarrow{C} = \overrightarrow{C}$, $\hat{I}\overrightarrow{D} = \overrightarrow{D}$; $\hat{\sigma}\overrightarrow{C} = \overrightarrow{D}$, $\hat{\sigma}\overrightarrow{D} = \overrightarrow{C}$.

Definition 2: For $n \geq 3$ agents, let each agent $i \in N$ possess a basic vector.

A vector is defined as the tensor product of n basic vectors:

$$\overrightarrow{\psi}_0 \equiv \overrightarrow{C}^{\otimes n} \equiv \underbrace{\overrightarrow{C} \otimes \cdots \otimes \overrightarrow{C}}_{n} \equiv \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}_{2^n \times 1} \tag{2}$$

The vector $\overrightarrow{C}^{\otimes n}$ contains n basic vectors $\overrightarrow{C}$ and 2^n elements. $\overrightarrow{C}^{\otimes n}$ is also denoted as $\overrightarrow{C \cdots CC}^n$. Similarly,

$$\overrightarrow{C \cdots CD}^n \equiv \underbrace{\overrightarrow{C} \otimes \cdots \otimes \overrightarrow{C}}_{n-1} \otimes \overrightarrow{D} = \begin{bmatrix} 0 \\ 1 \\ \cdots \\ 0 \end{bmatrix}_{2^n \times 1} \tag{3}$$

Obviously, there are 2^n possible vectors $\{\overrightarrow{C \cdots CC}^n, \cdots, \overrightarrow{D \cdots DD}^n\}$.

Definition 3: $\hat{J} \equiv \frac{1}{\sqrt{2}}(\hat{I}^{\otimes n} + i\hat{\sigma}^{\otimes n})$, *i.e.*,

$$\hat{J} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & & & & & i \\ & \cdots & & & \cdots & \\ & & 1 & i & & \\ & & i & 1 & & \\ & \cdots & & & \cdots & \\ i & & & & & 1 \end{bmatrix}_{2^n \times 2^n}, \hat{J}^{+} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & & & & & -i \\ & \cdots & & & \cdots & \\ & & 1 & -i & & \\ & & -i & 1 & & \\ & \cdots & & & \cdots & \\ -i & & & & & 1 \end{bmatrix}_{2^n \times 2^n} \tag{4}$$

where the symbol i denotes an imaginary number, and $\hat{J}^{+}$ is the conjugate transpose of $\hat{J}$. In what follows, we will not explicitly claim whether i is an imaginary number or an index. It is easy for the reader to know its exact meaning from the context.

Definition 4:

$$\overrightarrow{\psi}_1 \equiv \hat{J}\overrightarrow{\psi}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ i \end{bmatrix}_{2^n \times 1} \tag{5}$$

Definition 5: For $\theta \in [0, \pi]$, $\phi \in [0, \pi/2]$,

$$\hat{\omega}(\theta, \phi) \equiv \begin{bmatrix} e^{i\phi}\cos(\theta/2) & i\sin(\theta/2) \\ i\sin(\theta/2) & e^{-i\phi}\cos(\theta/2) \end{bmatrix}. \tag{6}$$

$\hat{\Omega} \equiv \{\hat{\omega}(\theta, \phi) : \theta \in [0, \pi], \phi \in [0, \pi/2]\}$. Hence, $\hat{I} = \hat{\omega}(0, 0)$, $\hat{\sigma} = i\hat{\omega}(\pi, 0)$. Let $\hat{\Omega}^0 = \{\hat{I}, \hat{\sigma}\}$.

Definition 6: For $j = 1, \cdots, n$, $\theta_j \in [0, \pi]$, $\phi_j \in [0, \pi/2]$, let $\hat{\omega}_j = \hat{\omega}(\theta_j, \phi_j)$,

$$\overrightarrow{\psi}_2 \equiv [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\overrightarrow{\psi}_1. \tag{7}$$

The dimension of $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$ is $2^n \times 2^n$. Since only two elements in $\overrightarrow{\psi}_1$ are non-zero, it is not necessary to calculate the whole $2^n \times 2^n$ matrix to yield $\overrightarrow{\psi}_2$. Indeed, we only need to calculate the leftmost and rightmost column of $[\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]$ to derive $\overrightarrow{\psi}_2$.

Definition 7: $\overrightarrow{\psi}_3 \equiv \hat{J}^+\overrightarrow{\psi}_2$.

Suppose $\overrightarrow{\psi}_3 = [\eta_1, \cdots, \eta_{2^n}]^T$, let $\Delta = [|\eta_1|^2, \cdots, |\eta_{2^n}|^2]$. It can be easily checked that $\hat{J}$, $\hat{\omega}_j$ ($j = 1, \cdots, n$) and $\hat{J}^+$ are all unitary matrices. Hence, $|\overrightarrow{\psi}_3|^2 = 1$. Thus, Δ can be viewed as a probability distribution, each element of which represents the probability that we randomly choose a vector from the set of all 2^n possible vectors $\{\overrightarrow{C \cdots CC}^n, \cdots, \overrightarrow{D \cdots DD}^n\}$.

Definition 8: Condition λ contains five parts. The first three parts are defined as follows:
λ_1: Given an SCR F, there exist two states $\hat{t}, \bar{t} \in \mathcal{T}$, $\hat{t} \neq \bar{t}$ such that $\hat{a} \succeq_i^{\bar{t}} \bar{a}$ (for each $i \in N$, $\hat{a} \in F(\hat{t})$, $\bar{a} \in F(\bar{t})$) with strict relation for some agent; and the number of agents that encounter a preference change around $\hat{a}$ in going from state $\hat{t}$ to $\bar{t}$ is at least two. Denote by l the number of these agents. Without loss of generality, let these l agents be the last l agents among n agents, *i.e.*, agent $(n - l + 1), \cdots, n$.
λ_2: Consider the state $\bar{t}$ specified in condition λ_1, if there exists another $\hat{t}' \in \mathcal{T}$, $\hat{t}' \neq \hat{t}$ that satisfies λ_1, then $\hat{a} \succeq_i^{\bar{t}} \hat{a}'$ (for each $i \in N$, $\hat{a} \in F(\hat{t})$, $\hat{a}' \in F(\hat{t}'))$ with

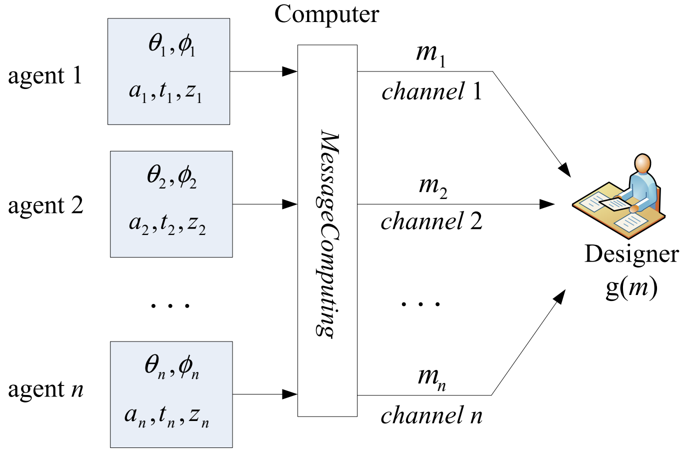

Fig. 1. The set-up of the agreement *ComplexMessage*. The algorithm *MessageComputing* is given in Definition 10.

strict relation for some agent.
λ_3: Consider the outcome $\hat{a}$ specified in condition λ_1, for any state $t \in \mathcal{T}$, $\hat{a}$ is top ranked for each agent i among the first $(n-l)$ agents.

3.3 An agreement that uses complex numbers

As we have seen, the Maskin's mechanism is an abstract mechanism. People seldom consider how the designer actually receives messages from agents. Roughly speaking, there are two manners: direct and indirect manner. In the former manner, the agents report their messages to the designer directly (e.g., orally, by hand, etc); in the latter manner, the agents report messages to the designer through some channels (e.g., Internet etc). In what follows, we assume the agents communicate with the designer through some channel.

Definition 9: Suppose conditions λ_1, λ_2 and λ_3 are satisfied, and the designer uses the Maskin's mechanism. An agreement *ComplexMessage* is constructed by the agents (see Fig. 1). It is constructed after the designer claims the outcome function g, and before agents send messages $m = (m_1, \cdots, m_n)$ to the designer. The algorithm *MessageComputing* is given in Definition 10.

Definition 10: The algorithm *MessageComputing* is defined as follows:
Input: $(\theta_i, \phi_i, a_i, t_i, z_i) \in [0, \pi/2] \times [0, \pi] \times A \times \mathcal{T} \times \mathbb{Z}_+$, $i = 1, \cdots, n$.
Output: $m_i \in A \times \mathcal{T} \times \mathbb{Z}_+$, $i = 1, \cdots, n$.
1: Reading (θ_i, ϕ_i) from each agent $i \in N$ (See Fig. 2(a)).
2: Computing the leftmost and rightmost columns of $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$ (See Fig. 2(b)).
3: Computing $\overrightarrow{\psi}_2 = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\overrightarrow{\psi}_1$, $\overrightarrow{\psi}_3 = \hat{J}^+\overrightarrow{\psi}_2$, and the probability distribution Δ (See Fig. 2(c)).

4: Randomly choosing a vector from the set of all 2^n possible vectors $\{\overrightarrow{C\cdots CC}^n, \cdots, \overrightarrow{D\cdots DD}^n\}$ according to the probability distribution Δ.
5: For each agent $i \in N$, let $m_i = (\hat{a}, \hat{t}, 0)$ (or $m_i = (a_i, t_i, z_i)$) if the i-th part of the chosen vector is $\overrightarrow{C}$ (or $\overrightarrow{D}$) (See Fig. 2(d)).
6: Sending $m = (m_1, \cdots, m_n)$ to the designer through channels $1, \cdots, n$.

Initially, in *ComplexMessage* all agents transfer their channels to the computer. After then, each agent $j \in N$ can leave his channel to the computer, or take back his channel and communicate with the designer directly:
1) Whenever any agent takes back his channel, every other agent will detect this deviation and take back their channels too, henceforth all agents will communicate with the designer directly.
2) When all agents leave their channels to the computer, the algorithm *MessageComputing* works, i.e., calculates $m = (m_1, \cdots, m_n)$ and sends it to the designer.
Put differently, after *ComplexMessage* is constructed, each agent $j \in N$ independently faces two options:
• $S(j, 0)$: leave his channel to the computer, and submit $(\theta_j, \phi_j, a_j, t_j, z_j)$ to the algorithm *MessageComputing*.
• $S(j, 1)$: take back his channel, and submit (a_j, t_j, z_j) to the designer directly.

Remark 1: Although the time and space complexity of *MessageComputing* are exponential, *i.e.*, $O(2^n)$, it works well when the number of agents is not large. For example, the runtime of *MessageComputing* is about 0.5s for 15 agents, and about 12s for 20 agents (MATLAB 7.1, CPU: Intel (R) 2GHz, RAM: 3GB).

Remark 2: The problem of Nash implementation requires complete information among all agents. In the last paragraph of Page 392 [2], Serrano wrote: "*We assume that there is complete information among the agents... This assumption is especially justified when the implementation problem concerns a small number of agents that hold good information about one another*". Hence, the fact that *MessageComputing* is suitable for small-scale cases (*e.g.*, less than 20 agents) is acceptable for Nash implementation.

Definition 11: Consider the state $\bar{t}$ specified in condition λ_1. Suppose λ_1 and λ_2 are satisfied, and $m = (m_1, \cdots, m_m)$ is computed by *MessageComputing*. $\$_{C\cdots CC}$, $\$_{C\cdots CD}$, $\$_{D\cdots DC}$ and $\$_{D\cdots DD}$ are defined as the payoffs to the n-th agent in state $\bar{t}$ when the chosen vector in Step 4 of *MessageComputing* is $\overrightarrow{C\cdots CC}^n$, $\overrightarrow{C\cdots CD}^n$, $\overrightarrow{D\cdots DC}^n$ or $\overrightarrow{D\cdots DD}^n$ respectively.

Note: When an agent faces a certain outcome, his payoff is the utility that he exactly obtains; when an agent faces an uncertain outcome among a set of outcomes, his payoff is the ex-ante utility before the final outcome is realized. It should be emphasized that the word "uncertain" is different from "random":

the latter means there is a certain probability distribution, whereas the former means the outcome is totally unknown before it is realized.

Definition 12: Suppose conditions λ_1 and λ_2 are satisfied. When the true state is $\bar{t}$, consider each message $m_i = (a_i, t_i, z_i)$, where a_i is top-ranked for each agent i. The rest parts of condition λ are defined as:
λ_4: $\$_{C\cdots CC} > \$_{D\cdots DD}$.
λ_5: $\$_{C\cdots CC} > \$_{C\cdots CD}\cos^2(\pi/l) + \$_{D\cdots DC}\sin^2(\pi/l)$.

3.4 Main result

Proposition 1: For $n \geq 3$, suppose the agents communicate with the designer through channels. Consider an SCR F that satisfies monotonicity and no-veto. Suppose the designer uses the Maskin's mechanism Γ and condition λ is satisfied, then in state $\bar{t}$ the agents can sign a self-enforcing agreement to make the Pareto-inefficient outcome $F(\bar{t})$ not be yielded in Nash equilibrium.

Proof: Since λ_1 and λ_2 are satisfied, then there exist two states $\hat{t}$, $\bar{t} \in \mathcal{T}$, $\hat{t} \neq \bar{t}$ such that $\hat{a} \succeq_i^{\bar{t}} \bar{a}$ (for each $i \in N$, $\hat{a} \in F(\hat{t})$, $\bar{a} \in F(\bar{t})$) with strict relation for some agent; and the number of agents that encounter a preference change around $\hat{a}$ in going from state $\hat{t}$ to $\bar{t}$ is at least two. Suppose the true state is $\bar{t}$. Let us check whether the agents can make the Pareto-inefficient outcome $\bar{a}$ not be implemented in Nash equilibrium by constructing *ComplexMessage*.

Firstly, note that after the agents construct *ComplexMessage*, the designer cannot discriminate whether a received message m_i is reported directly from agent i or sent by *MessageComputing*. Put differently, the timing steps of the designer are not changed:
Time 1: The designer claims the outcome function g to all agents;
Time 2: The designer receives $m = (m_1, \cdots, m_n)$;
Time 3: The designer computes the outcome $g(m)$.

Secondly, from the viewpoints of agents, the situation is changed. After constructing *ComplexMessage*, there are two possible cases:
1) Suppose every agent i chooses $S(i, 0)$, then the algorithm *MessageComputing* works. Consider the strategy profile chosen by the agents: each agent $i = 1, \cdots, (n-l)$ submits $(\theta_i, \phi_i) = (0, 0)$; each agent $i = (n-l+1), \cdots, n$ submits $(\theta_i, \phi_i) = (0, \pi/l)$. Since condition λ is satisfied, according to Lemma 1 (see Appendix), this strategy profile is a Nash equilibrium of Γ in state $\bar{t}$. As a result, in Step 4 of *MessageComputing*, the chosen vector will be $\overrightarrow{C\cdots CC}$; in Step 5 of *MessageComputing*, $m_i = (\hat{a}, \hat{t}, 0)$ for each $i \in N$. In the end, $g(m) = \hat{a} \notin F(\bar{t})$. Each agent i's payoff is $\$_{C\cdots CC}$.

2) Suppose some agent $i \in N$ chooses $S(i, 1)$, i.e., take back his channel and report m_i to the designer directly. Then all of the rest agents will take back their channels and report messages to the designer directly. Each agent i's payoff is $\$_{D \cdots DD}$.

Since condition λ_4 is satisfied, it is not profitable for any agent i to unilaterally take back his channel. According to Telser [3], *ComplexMessage* is a self-enforcing agreement among the agents. Put differently, although the agents collaborate to construct *ComplexMessage* between Time 1 and 2, they do not require a third-party to enforce *ComplexMessage*.

To sum up, in state $\bar{t}$, the agents can sign a self-enforcing agreement *ComplexMessage* to make the Pareto-inefficient outcome $\bar{a}$ not be implemented in Nash equilibrium. □

4 Conclusions

In this paper, we propose a self-enforcing agreement to help agents avoid the Pareto-inefficient outcome when they face a bad social choice rule. When the designer uses the Maskin's mechanism and receives messages from the agents through some communication channels (e.g., Internet), the designer cannot restrict the agents from signing such agreement. It should be noted that the introduction of complex numbers plays an important role in this paper. To the best of our knowledge, there is no similar work before. Since the Maskin's mechanism has been widely applied to many disciplines, there are many works to do in the future to generalize the self-enforcing agreement further.

References

[1] E. Maskin, Nash equilibrium and welfare optimality, *Rev. Econom. Stud.* **66** (1999) 23-38.

[2] R. Serrano, The theory of implementation of social choice rules, *SIAM Review* **46** (2004) 377-414.

[3] L.G. Telser, A theory of self-enforcing agreements. *Journal of Business* **53** (1980) 27-44.

[4] A.P. Flitney and L.C.L. Hollenberg, Nash equilibria in quantum games with generalized two-parameter strategies, *Phys. Lett. A* **363** (2007) 381-388.

Appendix

Lemma 1: Suppose the algorithm *MessageComputing* works. If condition λ is satisfied, consider the following strategy:
1) Each agent $i = 1, \cdots, (n-l)$ submits $(\theta_i, \phi_i) = (0, 0)$;
2) Each agent $i = (n-l+1), \cdots, (n-1)$ submits $(\theta_i, \phi_i) = (0, \pi/l)$;
then the optimal value of (θ, ϕ) for the n-th agent is $(0, \pi/l)$.

Proof: Since condition λ_1 is satisfied, then $l \geq 2$. Let

$$\hat{C}_l \equiv \hat{\omega}(0, \pi/l) = \begin{bmatrix} e^{i\frac{\pi}{l}} & 0 \\ 0 & e^{-i\frac{\pi}{l}} \end{bmatrix}_{2\times 2}, \qquad \text{thus, } \hat{C}_l \otimes \hat{C}_l = \begin{bmatrix} e^{i\frac{2\pi}{l}} & & & \\ & 1 & & \\ & & 1 & \\ & & & e^{-i\frac{2\pi}{l}} \end{bmatrix}_{2^2\times 2^2},$$

$$\underbrace{\hat{C}_l \otimes \cdots \otimes \hat{C}_l}_{l-1} = \begin{bmatrix} e^{i\frac{(l-1)}{l}\pi} & & & \\ & * & & \\ & & \cdots & \\ & & & e^{-i\frac{(l-1)}{l}\pi} \end{bmatrix}_{2^{l-1}\times 2^{l-1}}.$$

Here we only explicitly list the up-left and bottom-right entries because only these two entries are useful in the following discussions. The other entries in diagonal are simply represented as symbol $*$. Note that

$$\underbrace{\hat{I} \otimes \cdots \otimes \hat{I}}_{n-l} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \cdots & \\ & & & 1 \end{bmatrix}_{2^{n-l}\times 2^{n-l}},$$

thus,

$$\underbrace{\hat{I} \otimes \cdots \otimes \hat{I}}_{n-l} \otimes \underbrace{\hat{C}_l \otimes \cdots \otimes \hat{C}_l}_{l-1} = \begin{bmatrix} e^{i\frac{(l-1)}{l}\pi} & & & \\ & * & & \\ & & \cdots & \\ & & & e^{-i\frac{(l-1)}{l}\pi} \end{bmatrix}_{2^{n-1}\times 2^{n-1}}.$$

Suppose the n-th agent chooses arbitrary parameters (θ, ϕ) in his strategy

$(\theta, \phi, a_n, t_n, z_n)$, let

$$\hat{\omega}(\theta,\phi) = \begin{bmatrix} e^{i\phi}\cos(\theta/2) & i\sin(\theta/2) \\ i\sin(\theta/2) & e^{-i\phi}\cos(\theta/2) \end{bmatrix},$$

then,

$$\underbrace{\hat{I}\otimes\cdots\otimes\hat{I}}_{n-l}\otimes\underbrace{\hat{C}_l\otimes\cdots\otimes\hat{C}_l}_{l-1}\otimes\hat{\omega}(\theta,\phi)$$
$$= \begin{bmatrix} e^{i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) & * & & & & & \\ ie^{i\frac{(l-1)\pi}{l}}\sin(\theta/2) & * & & & & & \\ & & * & * & & & \\ & & * & * & & & \\ & & & & \cdots & & \\ & & & & & * & ie^{-i\frac{(l-1)\pi}{l}}\sin(\theta/2) \\ & & & & & * & e^{-i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) \end{bmatrix}_{2^n\times 2^n}.$$

Recall that

$$\overrightarrow{\psi}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ i \end{bmatrix}_{2^n\times 1},$$

thus,

$$\overrightarrow{\psi}_2 = [\underbrace{\hat{I}\otimes\cdots\otimes\hat{I}}_{n-l}\otimes\underbrace{\hat{C}_l\otimes\cdots\otimes\hat{C}_l}_{l-1}\otimes\hat{\omega}(\theta,\phi)]\overrightarrow{\psi}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} e^{i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) \\ ie^{i\frac{(l-1)\pi}{l}}\sin(\theta/2) \\ 0 \\ \cdots \\ 0 \\ -e^{-i\frac{(l-1)\pi}{l}}\sin(\theta/2) \\ ie^{-i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) \end{bmatrix}_{2^n\times 1},$$

$$\overrightarrow{\psi}_3 = \hat{J}^{+}\overrightarrow{\psi}_2 = \frac{1}{2}\begin{bmatrix} e^{i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) + e^{-i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) \\ ie^{i\frac{(l-1)\pi}{l}}\sin(\theta/2) + ie^{-i\frac{(l-1)\pi}{l}}\sin(\theta/2) \\ 0 \\ \cdots \\ 0 \\ e^{i\frac{(l-1)\pi}{l}}\sin(\theta/2) - e^{-i\frac{(l-1)\pi}{l}}\sin(\theta/2) \\ -ie^{i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) + ie^{-i[\frac{(l-1)\pi}{l}+\phi]}\cos(\theta/2) \end{bmatrix}_{2^n\times 1}$$

$$= \begin{bmatrix} \cos(\theta/2)\cos(\frac{l-1}{l}\pi+\phi) \\ i\sin(\theta/2)\cos\frac{l-1}{l}\pi \\ 0 \\ \cdots \\ 0 \\ i\sin(\theta/2)\sin\frac{l-1}{l}\pi \\ \cos(\theta/2)\sin(\frac{l-1}{l}\pi+\phi) \end{bmatrix}_{2^n\times 1} .$$

The probability distribution Δ is computed from $\overrightarrow{\psi}_3$:

$$P_{C\cdots CC} = \cos^2(\theta/2)\cos^2(\phi-\frac{\pi}{l}) \tag{8}$$

$$P_{C\cdots CD} = \sin^2(\theta/2)\cos^2\frac{\pi}{l} \tag{9}$$

$$P_{D\cdots DC} = \sin^2(\theta/2)\sin^2\frac{\pi}{l} \tag{10}$$

$$P_{D\cdots DD} = \cos^2(\theta/2)\sin^2(\phi-\frac{\pi}{l}) \tag{11}$$

Obviously,

$$P_{C\cdots CC} + P_{C\cdots CD} + P_{D\cdots DC} + P_{D\cdots DD} = 1.$$

Consider the payoff to the n-th agent,

$$\$_n = \$_{C\cdots CC}P_{C\cdots CC} + \$_{C\cdots CD}P_{C\cdots CD} + \$_{D\cdots DC}P_{D\cdots DC} + \$_{D\cdots DD}P_{D\cdots DD}. \tag{12}$$

Since λ_4 is satisfied, *i.e.*, $\$_{C\cdots CC} > \$_{D\cdots DD}$, then the n-th agent chooses $\phi = \pi/l$ to minimize $\sin^2(\phi - \frac{\pi}{l})$. As a result, $P_{C\cdots CC} = \cos^2(\theta/2)$.

Since λ_5 is satisfied, *i.e.*, $\$_{C\cdots CC} > \$_{C\cdots CD}\cos^2(\pi/l) + \$_{D\cdots DC}\sin^2(\pi/l)$, then the n-th agent prefers $\theta = 0$, which leads $\$_n$ to its maximum $\$_{C\cdots CC}$. Therefore, the optimal value of (θ, ϕ) for the n-th agent is $(0, \pi/l)$. □

Note: The proof of Lemma 1 is similar to the derivation of Eq. (25) [4].

```
% A Matlab program of the algorithm MessageComputing
start_time = cputime

% n: the number of agents. In Table 1, there are 3 agents: Apple, Lily, Cindy
n = 3;

% Defining the array of θ_i and φ_i, i = 1,…,n.
theta = zeros(n,1);
phi = zeros(n,1);

% Reading Apple's parameters. For example, ω̂_1 = ω̂_Apple = ω̂(0,π/2)
theta(1) = 0;
phi(1) = pi/2;

% Reading Lily's parameters. For example, ω̂_2 = ω̂_Lily = ω̂(0,π/2)
theta(2) = 0;
phi(2) = pi/2;

% Reading Cindy's parameters. For example, ω̂_3 = ω̂_Cindy = ω̂(0,0)
theta(3) = 0;
phi(3) = 0;
```

Fig. 2 (a). Reading each agent i's parameters θ_i and $\phi_i, i = 1, \cdots, n$.

```
% Defining two 2*2 matrices
A=zeros(2,2);
B=zeros(2,2);

% In the beginning, A represents ω̂_1
A(1,1)=exp(i*phi(1))*cos(theta(1)/2);
A(1,2)=i*sin(theta(1)/2);
A(2,1)=A(1,2);
A(2,2)=exp(-i*phi(1))*cos(theta(1)/2);
row_A=2;

% Computing ω̂_1 ⊗ ⋯ ⊗ ω̂_n
for agent = 2 : n
        % B varies from ω̂_2 to ω̂_n
        B(1,1) = exp(i*phi(agent))*cos(theta(agent)/2);
        B(1,2) = i*sin(theta(agent)/2);
        B(2,1) = B(1,2);
        B(2,2) = exp(-i*phi(agent))*cos(theta(agent)/2);

        % Computing the leftmost and rightmost columns of C= A ⊗ B
        C = zeros(row_A*2, 2);
        for row=1 : row_A
                C((row-1)*2+1, 1) = A(row,1) * B(1,1);
                C((row-1)*2+2, 1) = A(row,1) * B(2,1);
                C((row-1)*2+1, 2) = A(row,2) * B(1,2);
                C((row-1)*2+2, 2) = A(row,2) * B(2,2);
        end
        A=C;
        row_A = 2 * row_A;
end
% Now the matrix A contains the leftmost and rightmost columns of ω̂_1 ⊗ ⋯ ⊗ ω̂_n
```

Fig. 2 (b). Computing the leftmost and rightmost columns of $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$

```
% Computing $\vec{\psi}_2 = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}\vec{\psi}_0$
psi2 = zeros(power(2,n),1);
for row=1 : power(2,n)
        psi2(row) = (A(row,1) + A(row,2)*i) / sqrt(2);
end

% Computing $\vec{\psi}_3 = \hat{J}^{+}\vec{\psi}_2$
psi3 = zeros(power(2,n),1);
for row=1 : power(2,n)
        psi3(row) = (psi2(row) - i*psi2(power(2,n)-row+1)) / sqrt(2);
end

% Computing the probability distribution $\Delta$
distribution = psi3.*conj(psi3);
```

Fig. 2 (c). Computing $\vec{\psi}_2$, $\vec{\psi}_3$, Δ

```
% Randomly choosing a vector according to the probability distribution $\Delta$
random_number = rand;
temp = 0;
for index=1: power(2,n)
   temp = temp + distribution(index);
   if temp >= random_number
      break;
   end
end

% indexstr: a binary representation of the index of the chosen vector
%   '0' stands for $\vec{C}$ , '1' stands for $\vec{D}$
index_str = dec2bin(index-1);
sizeofindexstr = size(index_str);

% Defining an array of messages for all agents
m = cell(n,1);

% For each agent $i \in N$, the algorithm generates the message $m_i$
for index = 1 : n - sizeofindexstr(2)
   m{index,1} = strcat('s(',int2str(index),'): $\hat{a}$, $\hat{t}$, 0 ');
end
for index = 1 : sizeofindexstr(2)
   if index_str(index)=='0'      % Note: '0' stands for $\vec{C}$
      m{n-sizeofindexstr(2)+index,1} = strcat('s(',int2str(n-sizeofindexstr(2)+index),'): $\hat{a}$, $\hat{t}$, 0 ');
   else
      m{n-sizeofindexstr(2)+index,1} = strcat('s(',int2str(n-sizeofindexstr(2)+index),'):3rd , 4th ,5th parameters');
   end
end

% The algorithm sends messages $m_1, \cdots, m_n$ to the designer
for index = 1 : n
   disp(m(index));
end

end_time = cputime;
runtime=end_time – start_time
```

Fig. 2 (d). Computing all messages $m_1, \cdots, m_n$